

Relazione Progetto Programmazione per Dispositivi Mobili

E-Tutoring Flutter Application

Gruppo

Bortolotti Simone

Davide De Cenzo

Marignati Luca

Sommario

Sommario	2
Deviazioni rispetto a quanto specificato nella scheda dell'app	3
Metodologia di sviluppo utilizzata: metodologia agile	4
Funzionalità dell'app e design	6
Architettura dell'app: struttura del codice realizzato e di altre risorse realizzate	20
Sicurezza: Basic Authentication	20
Database MySql	21
Web Service implementati	24
Model	26
Paradigma MVC	27
Principali librerie utilizzate e le possibili alternative	28
flutter_secure_storage	29
Scelta implementativa	29
Implementazione	30
Alternative: Secure storage / Shared Preferences / SQLite / Local File Storage	31
Flutter e supporto per device multipli	32
Supporto per le lingue diverse	33
Test effettuati	35
test/controller - Mockito	35
test/screens	37

Deviazioni rispetto a quanto specificato nella scheda dell'app

La maggior parte delle funzionalità previste in fase di analisi sono state implementate con successo.

Le funzionalità non implementate sono:

- **ricezione via e-mail dell'iscrizione:** non ritenuta di fondamentale importanza in relazione alla quantità di lavoro richiesto per l'implementazione: occorre configurare un server SMTP per l'invio e la consegna dei messaggi di posta;
- **chat tra studenti e tutor:** per questioni di tempi e di consegne non è stata implementata la chat in quanto dopo un'analisi delle funzionalità dell'applicazione sarebbe costato troppo in relazione allo scopo dell'app, ovvero dare la possibilità agli studenti di prenotarsi per il tutoraggio dei corsi.
Inoltre, considerando che l'applicazione mostra il numero di telefono e l'e-mail del tutor, è possibile mettersi direttamente in contatto con il tutor stesso, senza aver bisogno di una chat interna all'applicazione.
Ipoteticamente, infatti, si può pensare che ulteriori esigenze dello studente vengano comunicate al tutor mediante chiamata e/o utilizzo di altre applicazioni (es. Skype, social network, WhatsApp, Telegram, ecc.).

Riguardo i vincoli temporali pianificati possiamo ritenerci soddisfatti del lavoro di pianificazione svolto. Per la data 1° luglio, lo sviluppo è stato terminato come previsto.

Le principali date previste per lo sviluppo dell'applicazione sono:

18 Maggio 2021	Inizio dello sviluppo
15 Giugno 2021	Prima Release con funzionalità minime
1 Luglio 2021	Release finale

Metodologia di sviluppo utilizzata: metodologia agile

È stata utilizzata una metodologia di sviluppo di tipo agile in modo da garantire una distribuzione continua di software efficienti creati in modo rapido e iterativo. La metodologia ci ha consentito di **rilasciare rapidamente release piccole, frequenti e modifiche al software in piccole porzioni** con l'obiettivo di migliorare la soddisfazione di un ipotetico cliente (sviluppo puramente accademico).

La metodologia ci ha consentito di adottare un approccio leggero e di integrare le modifiche in qualsiasi fase del ciclo di vita (planning – analisi – implementazione – testing – valutazione), anziché ostacolarle.



Sono state, prima, sviluppate le funzionalità strettamente necessarie:

- per esempio, la prima funzionalità implementata è stata quella della prenotazione di una lezione privata da parte di uno studente;
- successivamente sono state, invece, implementate funzionalità aggiuntive come, ad esempio, il calendario, la possibilità di cambiare la password, la possibilità di effettuare una recensione di un tutor, ecc.

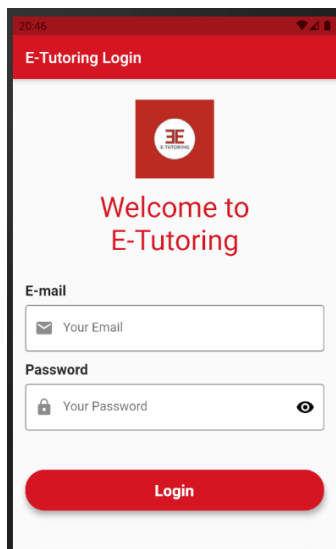
In particolare, come team di sviluppo abbiamo lavorato mettendo in risalto i seguenti valori fondamentali:

- **comunicazione**: come gruppo siamo rimasti quotidianamente in contatto per creare un continuo flusso e scambio di informazioni in modo da velocizzare lo sviluppo e la risoluzione di problemi;
- **pair programming (programmazione in coppia)**: il codice è stato prodotto dal team (mediante condivisione di schermo) in contemporanea sullo stesso task/funzione. Ci siamo alternati per eseguire la fase di scrittura e di revisione del codice (conducente e osservatore) scambiandoci i ruoli;
- **semplicità**: abbiamo sviluppato l'applicazione seguendo il principio **"il modo più semplice per raggiungere un risultato è anche quello più efficace"**. Abbiamo, dunque, sviluppato l'applicazione privilegiando la semplicità dei concetti (progettazione della logica e delle view).
- **modifiche incrementali**: le modifiche sono state eseguite gradualmente: invece di attuare grandi aggiornamenti, sono state implementate le funzionalità e affrontati i problemi uno dopo l'altro fino alla completa risoluzione;
- **processo continuo di refactoring**: il progetto è stato continuamente migliorato (in maniera iterativa) rimuovendo codice duplicato, fattorizzando il codice utilizzato in un classi/widget (es. definendo funzioni di libreria e variabili globali) e rimuovendo gli elementi superflui;
- **simple design**: è stato seguito questo principio per permettere di avere un codice comprensibile a tutti. Si è cercato di separare le varie componenti seguendo il pattern MVC (Model – View – Controller):
 - le chiamate http sono state implementate in opportune classi/funzioni in modo da separare la logica dalla visualizzazione (view);
 - i modelli che rappresentano i campi di una tabella/oggetto relativi al backend (WS/DB) sono stati implementati in classi che rappresentano i singoli oggetti;

- **testing:** Test-Driven Development: il team di sviluppo ha scritto i test prima ancora di creare il codice sorgente definitivo: in particolare i test maggiormente implementati sono stati quelli che riguardano l'utilizzo del WS. Prima di procedere all'implementazione dei metodi utilizzati per rappresentare i dati nei vari Widget sono stati ampiamente testate le chiamate http utilizzando Mockito e verificando che gli oggetti JSON restituiti fossero realmente quelli attesi;
- **feedback continuo:** abbiamo testato il codice e la funzione a livello di view fin dal primo giorno in modo da verificare la corretta implementazione;
- **software funzionante più che documentazione esaustiva.**

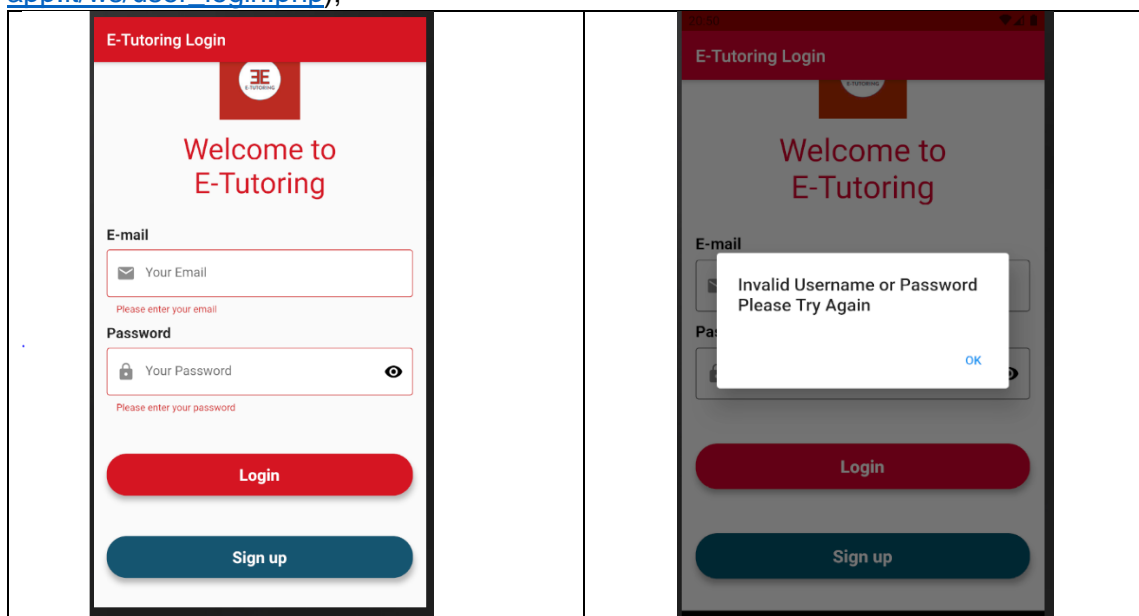
Funzionalità dell'app e design

Login: Avviando l'applicazione ci imbattiamo nella schermata iniziale (E-Tutoring Login) dove gli utenti possono eseguire l'accesso all'applicazione inserendo e-mail e password, oppure se utilizzano l'app per la prima volta, hanno la possibilità di iscriversi;

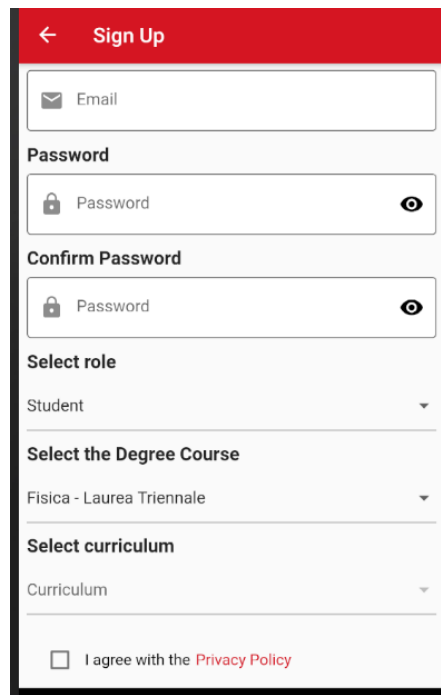


È stata prevista una doppia validazione:

- **FRONT-END:**
 - o e-mail e password non vuoti;
 - o e-mail che rispetta il formato "test@email.it";
- **BACK-END:** verifica che l'email e la password inserite dall'utente corrispondano a quelle inserite nel DB (chiamando il WS user_login.php - https://www.e-tutoring-app.it/ws/user_login.php);



1. **Registrazione di un nuovo utente:** un utente GUEST può registrarsi come studente o tutor all'applicazione compilando i campi relativi all'email, alla password, e selezionando il ruolo, il grado del corso e il curriculum;



A mobile application sign-up form with a red header bar containing a back arrow and the text "Sign Up". The form contains several input fields and dropdown menus. The "Email" field has an envelope icon. The "Password" and "Confirm Password" fields have a lock icon and a toggle eye icon. The "Select role" dropdown shows "Student". The "Select the Degree Course" dropdown shows "Fisica - Laurea Triennale". The "Select curriculum" dropdown shows "Curriculum". At the bottom, there is a checkbox and the text "I agree with the Privacy Policy".

Sign Up

Email

Password

Confirm Password

Select role

Student

Select the Degree Course

Fisica - Laurea Triennale

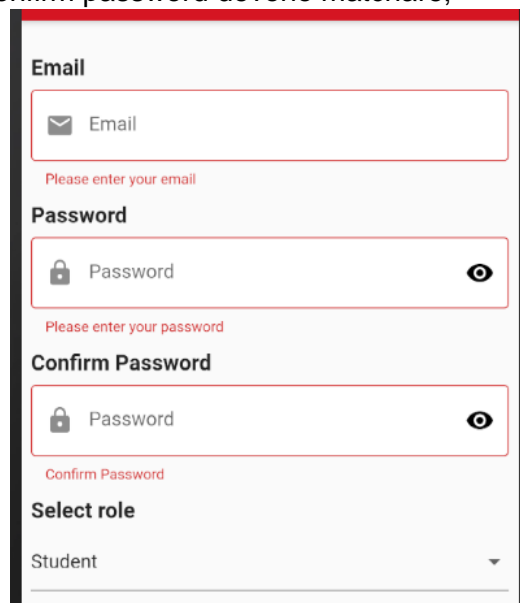
Select curriculum

Curriculum

☐ I agree with the [Privacy Policy](#)

Filtri e validazioni lato front-end:

- e-mail e password non vuoti;
- e-mail che rispetta il formato "test@email.it";
- password e confirm password devono matchare;



The same sign-up form as above, but with red borders around the input fields and red error messages below them. The "Email" field has the error "Please enter your email". The "Password" field has the error "Please enter your password". The "Confirm Password" field has the error "Confirm Password".

Email

Email

Please enter your email

Password

Password

Please enter your password

Confirm Password

Password

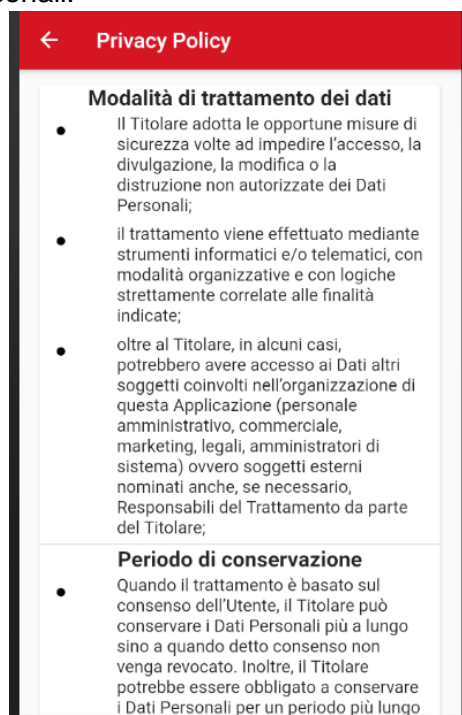
Confirm Password

Select role

Student

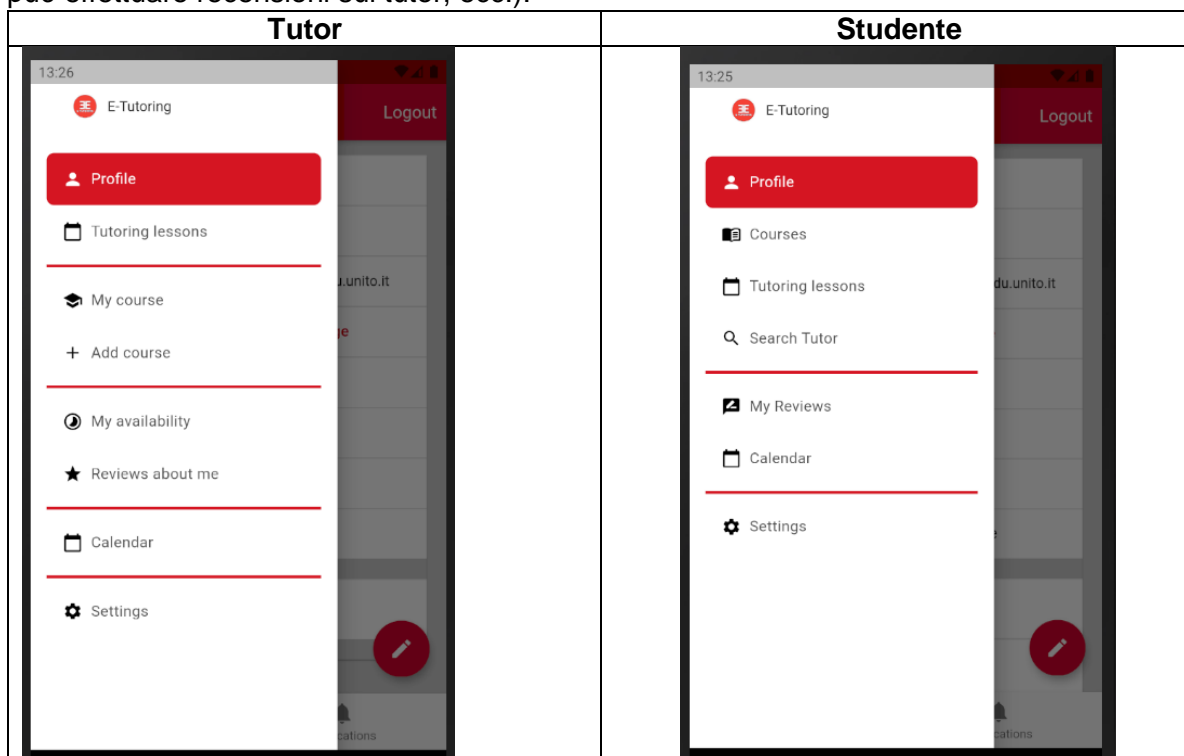
2. Privacy policy

Nella Schermata seguente vengono descritte all'utente le modalità di trattamento dei dati, come ad esempio le misure di sicurezza utilizzate per impedire la divulgazione di dati personali e a chi è possibile dare l'accesso ad essi. Inoltre informa l'utente sul periodo di conservazione dei dati personali.



3. Drawer menu

Tramite una sorta di Access Control List è stata differenziata la visibilità del menù in base al ruolo dell'utente. Alcune funzionalità sono comuni ad entrambi i profili, altre sono specifiche per il tipo di utente (es. il tutor può aggiungere delle disponibilità di orario/giorni, lo studente può effettuare recensioni sui tutor, ecc.).



4. Home differente per profilo studente e tutor

La schermata di profilo è divisa in due TAB:

- profilo;
- notifiche;

Tab profilo

Nelle schermate seguenti vengono mostrate le differenze tra il profilo studente (a destra) ed il profilo tutor (a sinistra). Da notare che la schermata viene “scomposta” in due sezioni separate: la prima riguarda i dati personali dell’utente ed è identica sia per lo studente che per il tutor; la seconda parte identifica i dati dello studente o del tutor e vediamo che differiscono: per il tutor viene indicato il ruolo, mentre per lo studente oltre che al ruolo vengono indicate anche altre informazioni come, ad esempio, il numero di matricola e le informazioni riguardi il corso di Laurea a cui lo studente è iscritto;

Tutor	Studente

Inoltre, in questo widget sono presenti le funzionalità per effettuare il Logout (in alto a destra) e tramite il floatingActionButton (Icons.edit) è possibile fare la modifica dei dati inseriti dall’utente.

Tab notifiche

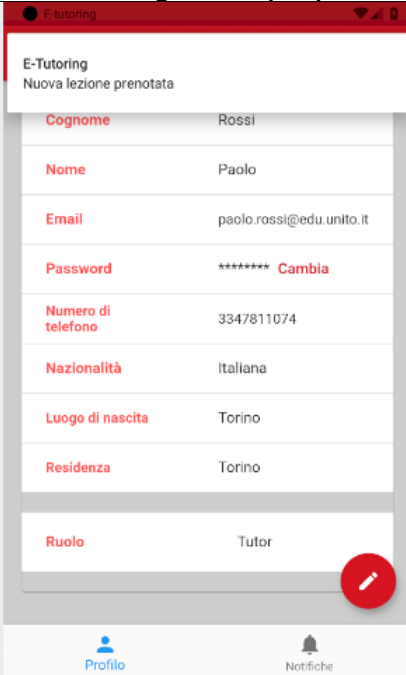
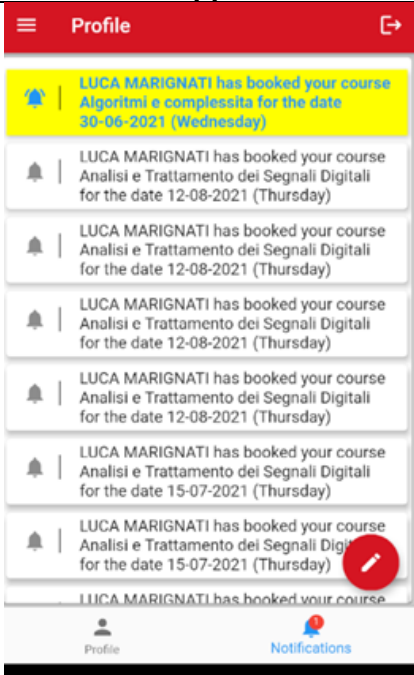
In questo tab è possibile visionare le notifiche dei Tutor.

Non abbiamo previsto notifiche per gli studenti.

I tutor ricevono una notifica quando uno studente si iscrive ad uno slot relativo ad un corso. Es. Davide De Cenzo si iscrive al corso di Programmazione per Dispositivi Mobili tenuto da Paolo Rossi il giorno 20 Luglio 2020 dalle ore 16:00 alle ore 19:00.

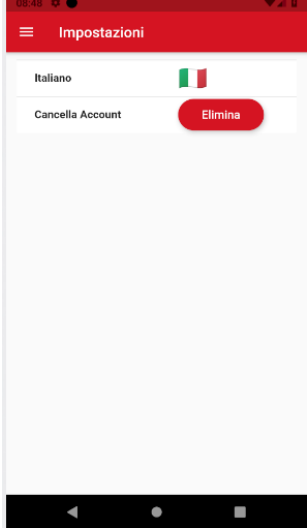
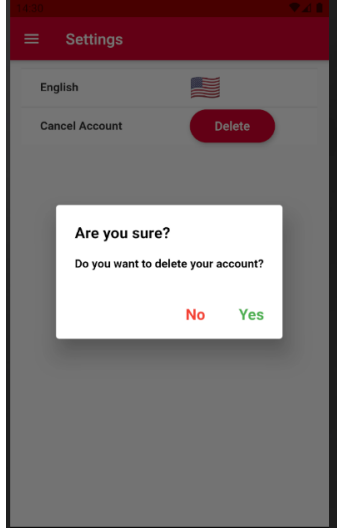
In questo caso, il tutor riceverà la notifica sia a livello globale (utilizzando Firebase per la gestione e l'invio delle notifiche) di sistema operativo (es. Android) sia a livello locale di applicazione:

- incremento del contatore delle nuove notifiche;
- visualizzazione evidenziata delle nuove notifiche (in giallo);

Notifica a livello globale (SO)	Notifica locale all'applicazione
	

5. Settings/Impostazioni

Il Widget delle impostazioni è molto semplice e minimale per garantire un corretto e veloce funzionamento da parte dell'utente. È possibile visualizzare la lingua (supporta Italiano ed Inglese), ed è possibile eliminare l'account dopo aver confermato di voler realmente eseguire l'operazione di cancellazione (AlertDialog).

	
---	--

TUTOR

6. Funzionalità del Tutor: Tutoring Lesson/lezioni private

In questo Widget è possibile visualizzare la lista (ListView) delle lezioni che il tutor deve sostenere.

Per ogni lezione vengono indicati:

- il nome del corso;
- il nome dello studente;
- il giorno di prenotazione;
- l'orario di prenotazione.

Per facilitare l'usabilità del widget è stata prevista una **funzionalità di ricerca** (in alto a destra) in cui il tutor può ricercare le proprie lezioni private in base al nome del corso.

In basso a destra, tramite il floatingActionButton (TODAY) è possibile visionare le lezioni che il tutor deve sostenere il giorno corrente in modo da sapere sempre se vi sono lezioni prenotate da sostenere oggi ed evitare dimenticanze.

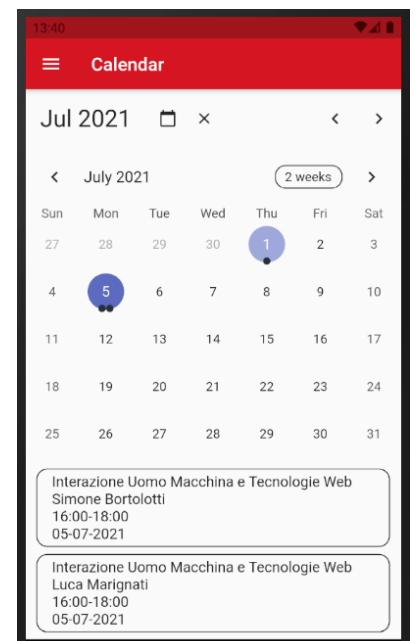
Proprio per questo, si è deciso di implementare un **Widget Calendario** interno all'applicazione che mostra le lezioni private (vedi sotto).



7. Funzionalità del Tutor: Calendario

Nella sezione sottostante possiamo visionare il calendario personale per ogni tutor con i relativi impegni, ossia con le relative lezioni private ad una certa data in una certa ora. Per semplificarne la lettura sul calendario attraverso dei punti è possibile capire se in un determinato giorno sono previste lezioni o meno.

Cliccando sul giorno è possibile anche visionare più nel dettaglio le lezioni relative a quel giorno



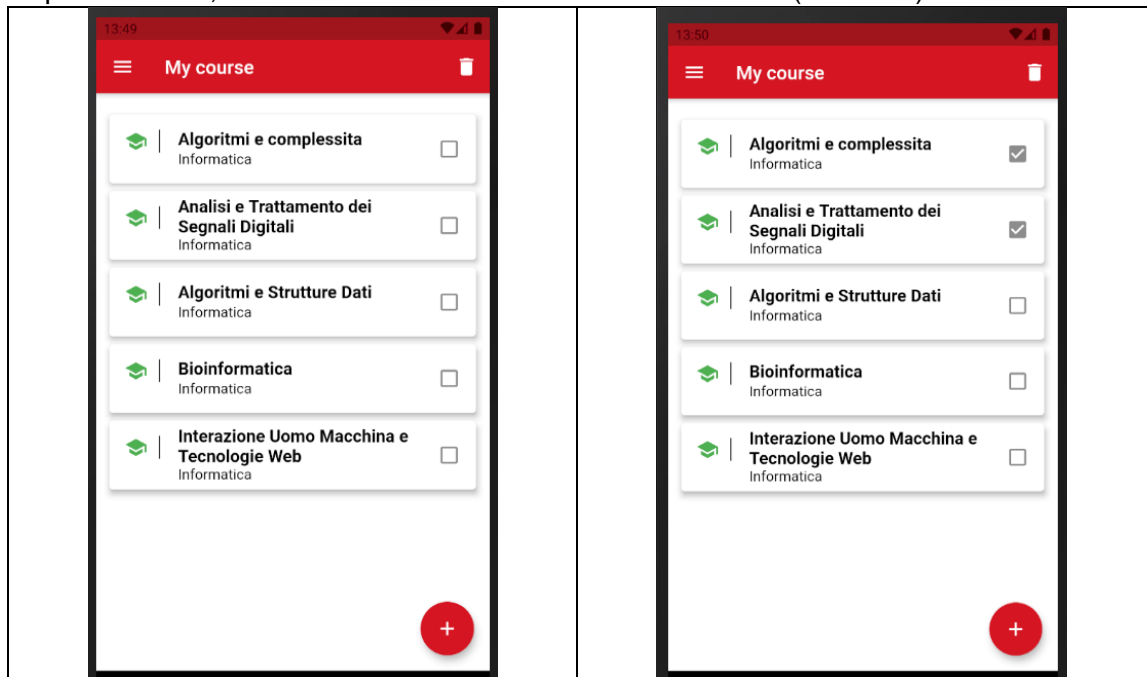
8. Funzionalità del Tutor: My Course/I miei corsi

Questo Widget offre la possibilità di visualizzare la lista (ListView) dei corsi insegnatisi dal tutor.

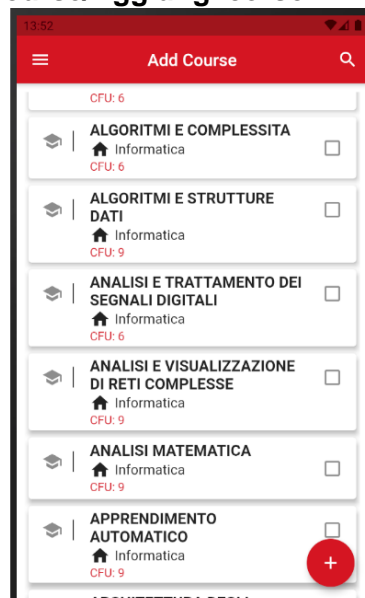
In aggiunta è possibile:

- aggiungere un corso cliccando il floatingActionButton in basso a destra: si rimanda al Widget AddCourse;
- eliminare i corsi su cui offrire le proprie ripetizioni agli studenti: la listView è selezionabile, ovvero è possibile selezionare i corsi che si vogliono eliminare.

Dopo la lezione, occorre cliccare sul bottone in alto a destra (il cestino).



9. Funzionalità del Tutor: Add Course/Aggiungi corso



Tramite questo widget, il tutor può aggiungere dei corsi a quelli da lui insegnati.

Come si nota dalla figura sopra, è possibile vedere la lista (ListView) di tutti i corsi.

Tale ListView è selezionabile, il tutor può selezionare i corsi che vuole aggiungere cliccando sul floatingActionButton in basso a destra (Icon.add).

Inoltre, per migliorare l'usabilità è stata prevista la funzionalità di ricerca (in alto a destra).

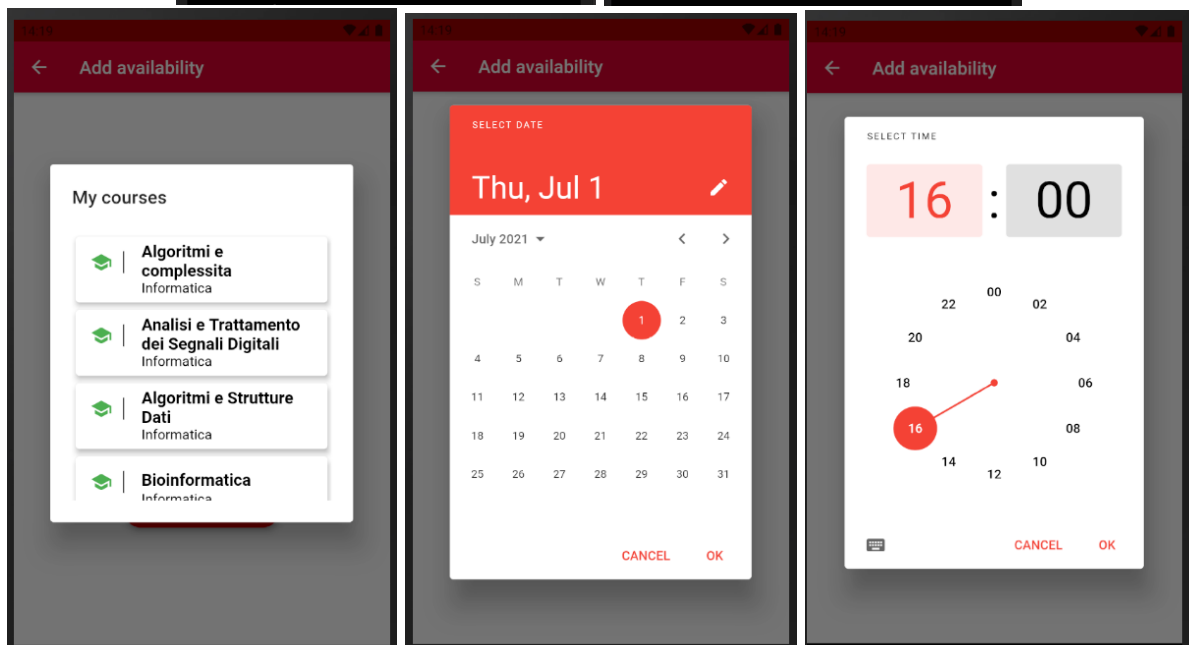
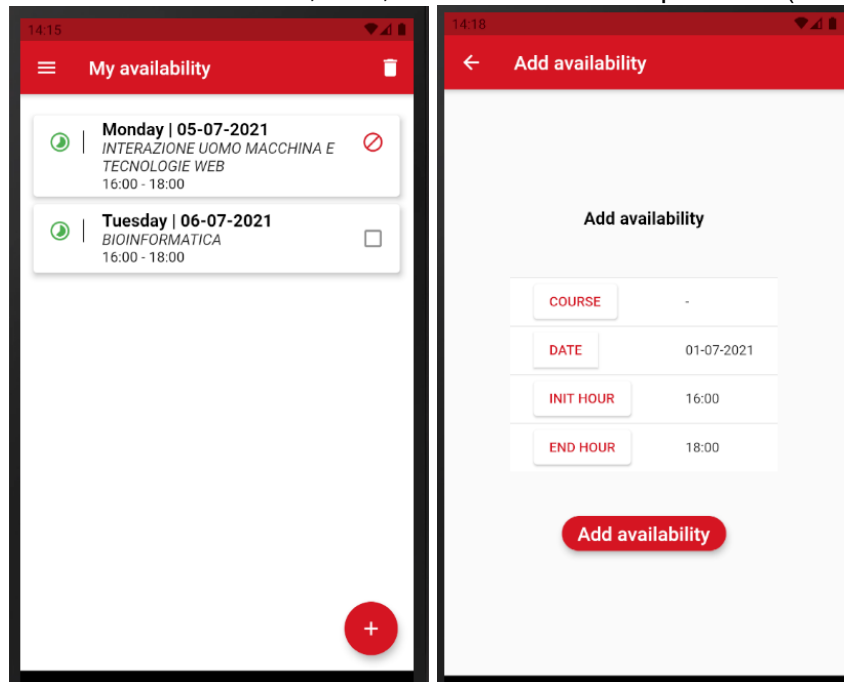
10. Funzionalità del Tutor: My Availability/Le mie disponibilità

In questo Widget è possibile visualizzare la lista delle disponibilità del tutor.

Come si nota, il singolo item della ListView mostra se è selezionabile o bloccato se la lezione è stata già prenotata da uno studente o meno.

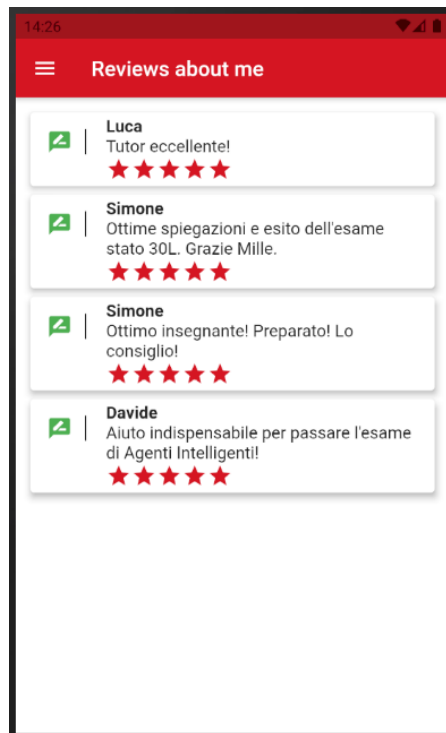
Nel caso in cui non è stata prenotata da nessuno ed è selezionabile è possibile rimuovere la disponibilità selezionando lo slot e cliccando sul cestino (in alto a destra).

In basso a destra è presente il floatingActionButton (Icons.add) che permette di aggiungere una disponibilità selezionando corso, data, inizio e fine della disponibilità (vedi figure sotto).



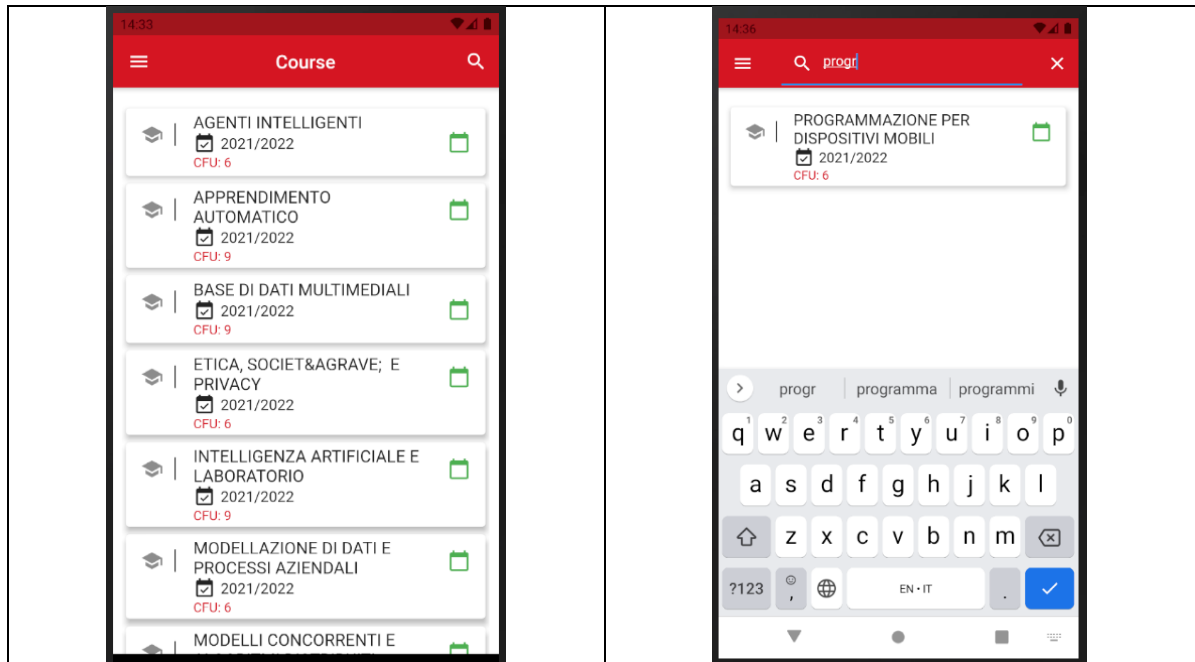
11. Funzionalità del Tutor: Review about me/recensioni su di me

In questo Widget è possibile visualizzare la lista (ListView) delle recensioni fatte dagli studenti. Ogni recensione mostra il nome dello studente, il suo commento sul tutor e un voto (da 1 a 5).



STUDENTE

12. Funzionalità dello Studente: Course/Corsi



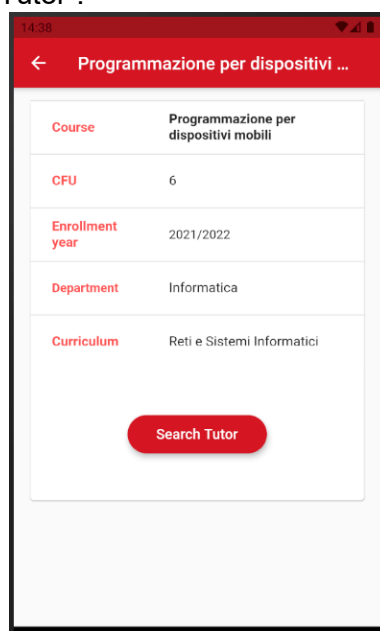
Il Widget mostra i corsi che possono essere scelti per poter richiedere una lezione privata. Viene mostrata una ListView contenente tutti i corsi relativi al corso di Laurea a cui l'utente è iscritto (Es. laurea Magistrale di Informatica). La nostra applicazione comprende tutti i corsi relativi all'università di informatica, matematica e fisica (laurea triennale o magistrale).

Il Widget offre la possibilità di ricercare un corso in base al nome.

13. Funzionalità dello Studente: Course Detail/ Dettaglio del corso

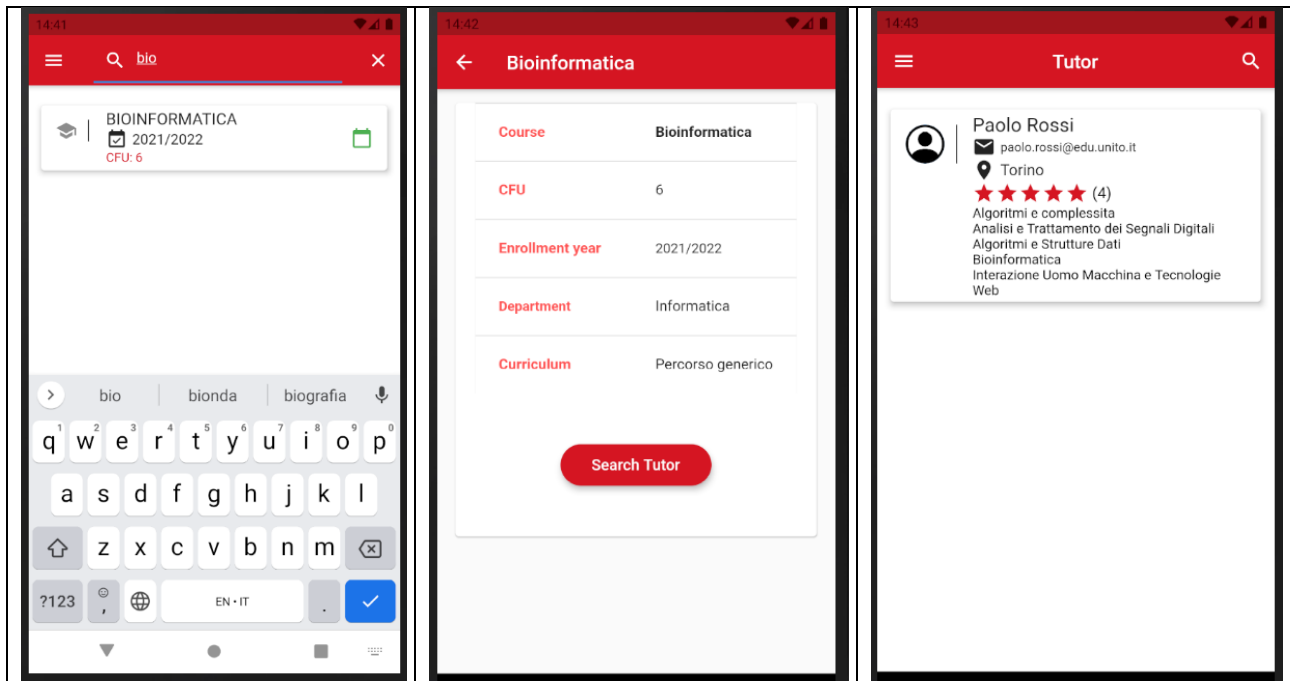
Il click di un item della ListView del Widget sopra citato (Course) permette di visualizzare le informazioni di dettaglio del corso (CFU, Dipartimento, Currisulum).

In questo Widget è possibile procedere alla ricerca del tutor in base al corso selezionato cliccando sul bottone "Search Tutor".



Esempio di interazione dell'utente:

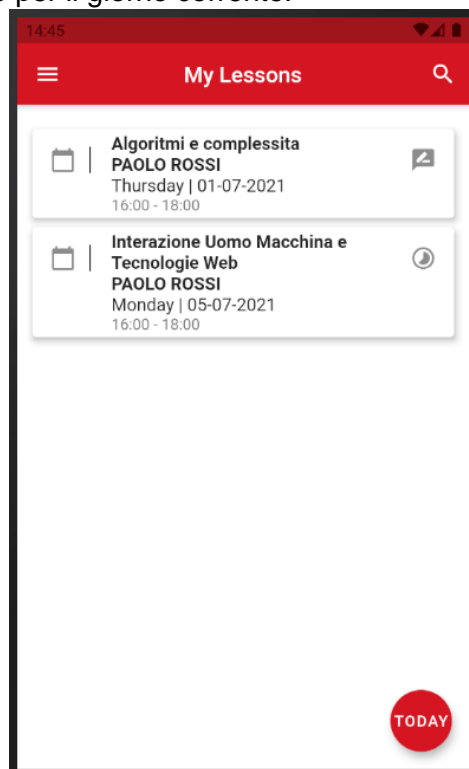
- Ricerca del corso "Bioinformatica"
- Visualizzazione del dettaglio del corso: ricerca Tutor
- Ricerca dei Tutor che offrono ripetizioni del corso selezionato



14. Funzionalità dello Studente: My Lesson/Le mie lezioni

In maniera simile a quanto visto per le lezioni offerte dai Tutor, uno Studente può visualizzare la lista delle lezioni prenotate in ordine di data crescente.

Anche qui, per facilitare l'usabilità, è possibile ricercare una lezione specifica e visualizzare se vi sono lezioni prenotate per il giorno corrente.

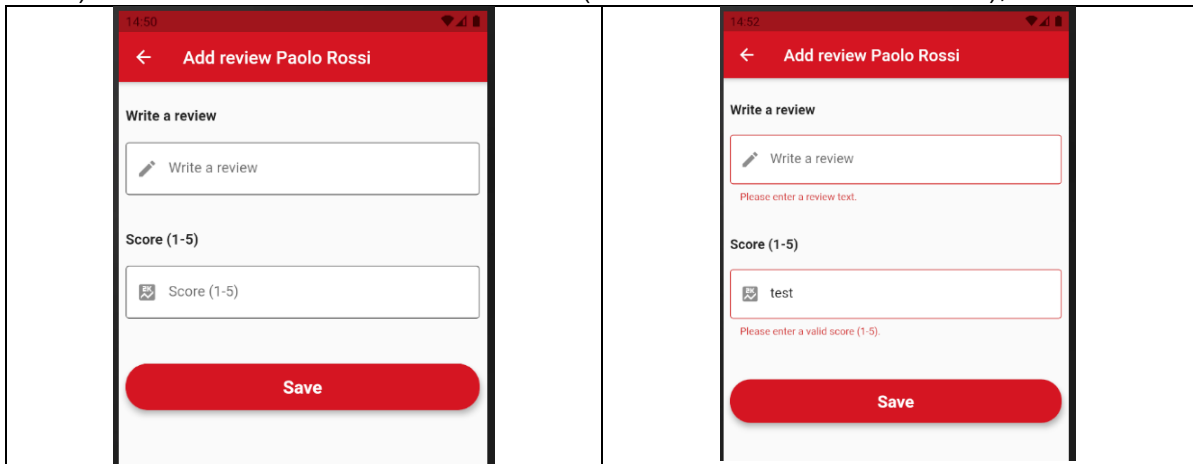


15. Funzionalità dello Studente: Add review/fare una recensione

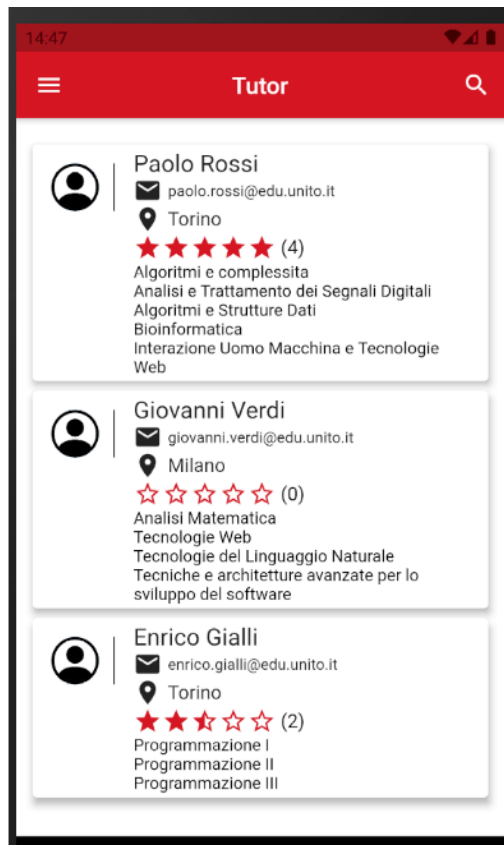
Come si nota dalla lista precedente, vi sono 2 tipi di item della ListView:

1. lezioni sostenute che possono essere recensite da parte dello studente (lezioni con data minore a data corrente);
2. lezioni ancora da sostenere (quindi non recensibili).

Le lezioni sostenute possono essere recensite dallo studente cliccando sull'item specifica. Al click dell'item, si ha il Widget che permette di scrivere una recensione al Tutor (es. Paolo Rossi) definendo un commento e uno score (intero da 1 a 5 con validazione);



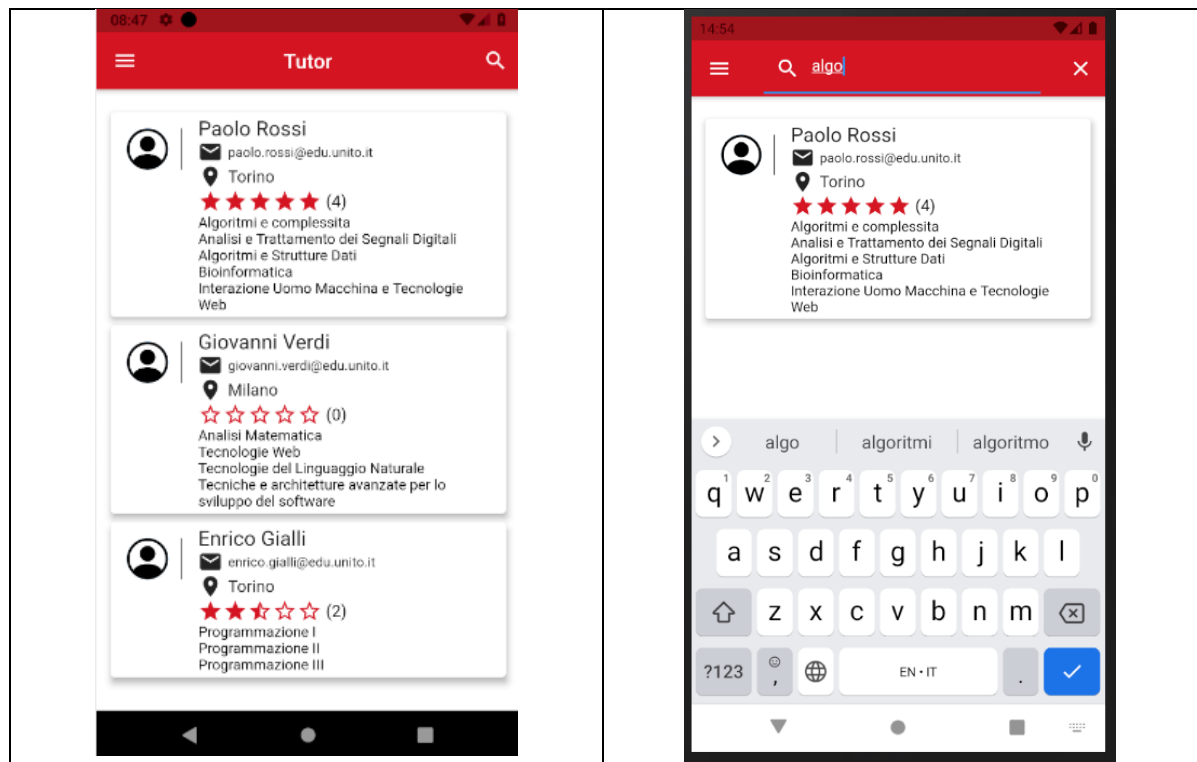
16. Funzionalità dello Studente: Search Tutor/Ricerca Tutor



Questo Widget fornisce la lista (ListView) dei tutor disponibili.

Ogni elemento della lista visualizza le informazioni relative al singolo tutor come ad esempio il nome, l'e-mail, i suoi insegnamenti e la media degli score delle recensioni fatte dagli studenti.

Anche qui è possibile ricerca i tutor per nome e insegnamenti.

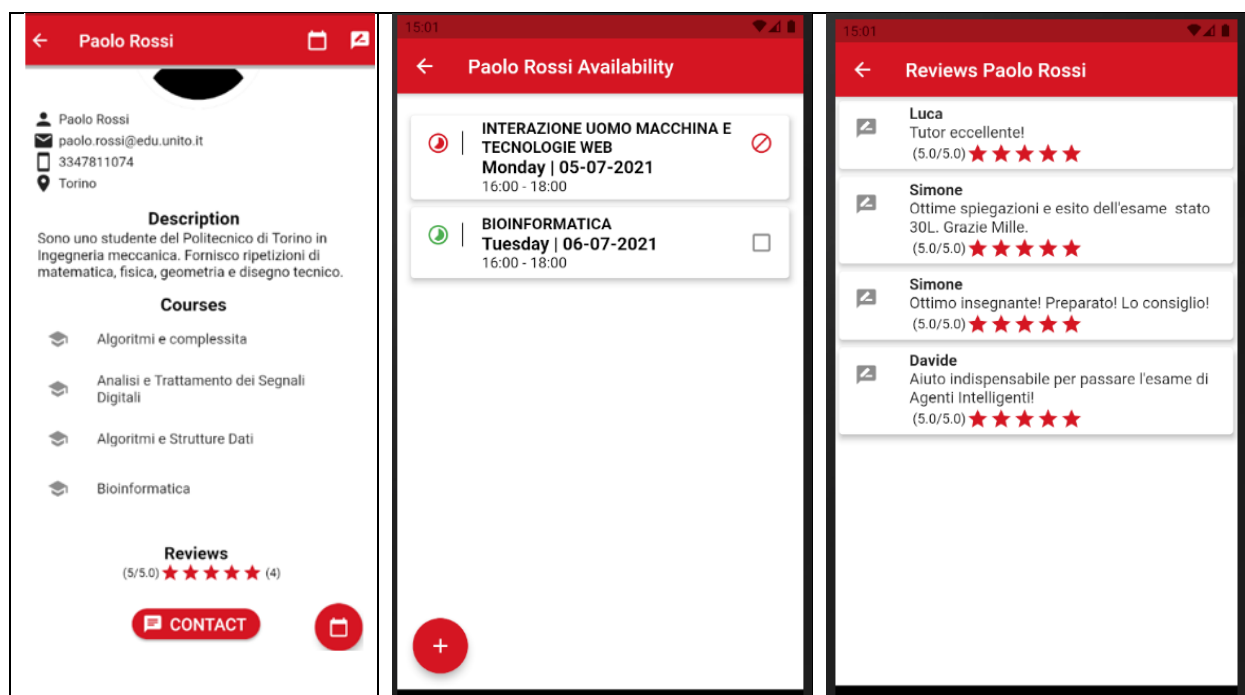


17. Funzionalità dello Studente: Dettaglio del Tutor

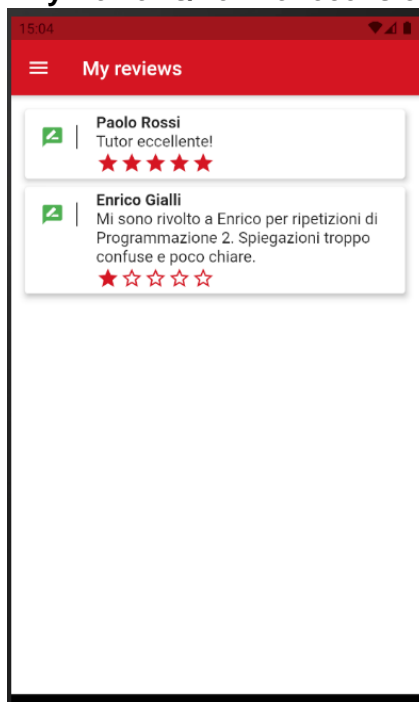
Questo Widget visualizza le informazioni del tutor scelto: informazioni generali, descrizione, corsi insegnati e media delle recensioni.

Inoltre, sono fornite le segue funzionalità:

- floatingActionButton icons.calendar che permette di visualizzare le disponibilità del tutor;
- icons.review che permette di visualizzare le recensioni fatte dagli studenti.



20. Funzionalità dello Studente: My Reviews/Le mie recensioni



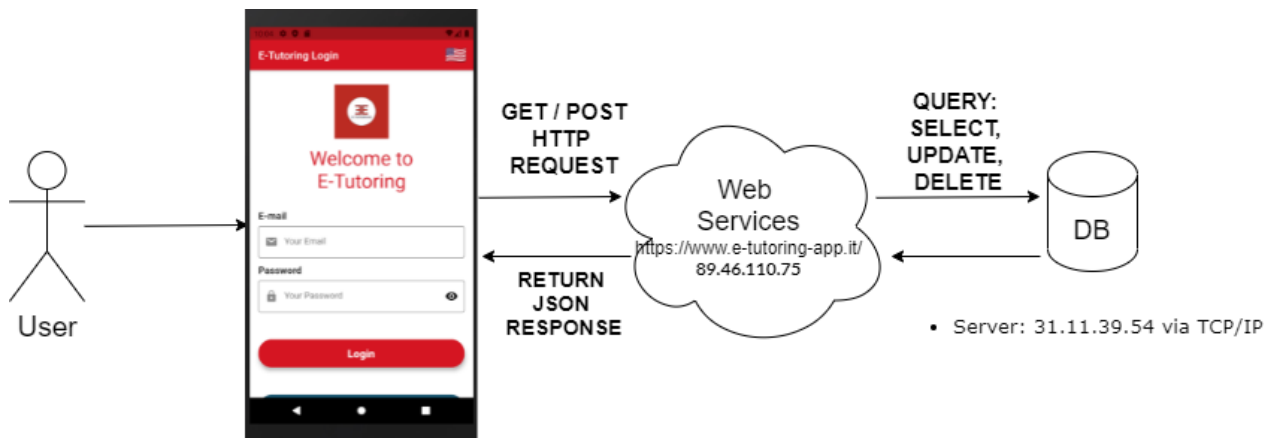
21. Funzionalità dello Studente: Calendario

Il Widget mostra le lezioni prenotate dallo studente.



Architettura dell'app: struttura del codice realizzato e di altre risorse realizzate

CLIENT → WEB SERVICES → DB MYSQL



1. l'utente interagisce con l'applicazione tramite un client (dispositivo, applicazione IOS o Android);
2. i dati provengono dal back-end:
 - a. il client effettua chiamate http (GET/POST method) richiamando i web services implementati;
 - b. i WS effettuano query (select) oppure operazioni di write (update o delete) sui dati contenuti nel DB MySQL;
 - c. i WS restituiscono, in formato JSON, i dati richiesti oppure l'esito dell'operazione richiesta (es. login, signup, delete profile, ecc.).

Per semplicità abbiamo provveduto a pubblicare in un dominio vero e proprio i WS sviluppati in modo da testare il reale funzionamento anche da dispositivo fisico e non solo mediante emulatore.

Dominio: <https://www.e-tutoring-app.it/89.46.110.75>

Sicurezza: Basic Authentication

I dati esposti contengono delle informazioni riservate (es. e-mail, indirizzo, ruolo dell'utente, ecc.) e, dunque, per accedere ai WS che comunicano con il database MySQL è stata predisposta una forma di autenticazione (**basic authentication**).

e-tutoring-app.it/ws/

Accedi

https://www.e-tutoring-app.it

Nome utente

Password

Accedi

Annulla

Database MySql

e_tutoring user id : int(20) username : varchar(70) password : varchar(40) email : varchar(50) created_at : datetime updated_at : datetime	e_tutoring role role_id : int(11) role_name : varchar(20) role_description : varchar(99)	e_tutoring course_path_degree course_id : int(11) degree_path_id : int(11) degree_id : int(11)	e_tutoring degree_type degree_type_id : int(20) degree_type_name : varchar(30) degree_type_note : text
e_tutoring user_attribute user_attribute_id : int(11) firstname : varchar(50) lastname : varchar(50) description : text img : varchar(99) badge_number : int(11) cf : varchar(30) birth_date : date birth_city : varchar(50) residence_city : varchar(50) address : varchar(50) nationality : varchar(50) gender : varchar(5) phone_number : varchar(20) degree_id : int(11) degree_path_id : int(11) role_id : int(11)	e_tutoring tutor_course tutor_course_id : int(11) note : text user_id : int(11) course_id : int(11)	e_tutoring degree degree_id : int(20) degree_name : varchar(30) degree_cfu : int(11) degree_description : text degree_type_id : int(11) degree_location : varchar(99) degree_athenaeum : varchar(99)	e_tutoring degree_path degree_path_id : int(11) degree_path_name : varchar(99) degree_path_description : varchar(99) degree_path_note : text degree_id : int(11)
e_tutoring tutor_time_slot tutor_time_slot_id : int(11) day : varchar(20) hour_from : varchar(5) hour_to : varchar(5) user_id : int(11)	e_tutoring private_lesson private_lesson_id : int(11) private_lesson_start_hour : varchar(10) private_lesson_end_hour : varchar(10) private_lesson_day : varchar(50) private_lesson_location : varchar(50) private_lesson_note : text user_id : int(11)	e_tutoring review review_id : int(11) user_tutor_id : int(11) user_id : int(11) review_star : int(11) review_comment : text	e_tutoring course course_id : int(11) course_name : varchar(99) course_cfu : int(11) enrollment_year : varchar(20) study_year : int(5) teaching_type : varchar(99) dac : varchar(10) department : varchar(30) curriculum : varchar(99) ssd : varchar(99) delivery_mode : varchar(30) language : varchar(30) didactic_period : varchar(30) component_type : varchar(99)

- **user**: contiene le informazioni utilizzate per il login all'applicazione (username, password, email). La password, ovviamente, non viene salvata in chiaro ma viene mantenuta in forma cryptata (md5).

Esempio:

id	username	password	email	created_at	updated_at
1	luca.marignati	098f6bcd4621d373cade4e832627b4f6	luca.marignati@edu.unito.it	2021-05-07 09:15:35	2021-05-07 09:15:35
2	simone.bortolotti	098f6bcd4621d373cade4e832627b4f6	simone.bortolotti@edu.unito.it	2021-05-07 09:15:35	2021-05-07 09:15:35
3	davide.decenzo	098f6bcd4621d373cade4e832627b4f6	davide.decenzo@edu.unito.it	2021-05-07 09:15:35	2021-05-07 09:15:35

- **user_attribute**: contiene gli attributi dell'utente (es. nome, cognome, numero di matricola, città di residenza ecc.).

Chiavi esterne:

- o **user_id**: si riferisce all'utente, tramite l'identificativo (attributo id), della tabella user;
- o **role_id**: si riferisce al ruolo dell'utente (Studente o Tutor) tramite l'identificativo role_id della tabella role;
- o **degree_path_id**: si riferisce al curriculum scelto in fase di registrazione dell'utente. L'attributo in questione si riferisce all'attributo degree_path_id della tabella degree_path (chiave esterna);
- o **degree_id**: si riferisce al corso di laurea a cui l'utente è iscritto.

Esempio:

user_attribute_id firstname lastname

1	Luca	Marignati
---	------	-----------

phone_number degree_id degree_path_id role_id user_id

3347811074	2	1	1	1
------------	---	---	---	---

- **role:** contiene i ruoli che gli utenti possono assumere nell'applicativo;

role_id **role_name** **role_description**

1	Student	Student
2	Tutor	Tutor
3	Tutor/Student	Tutor & Student

- **course:** contiene le informazioni relative ai corsi (es. nome, CFU, tipo di insegnamento, SSD, ecc.);

course_id	course_name	course_cfu	enrollment_year <small>Academic year of enrollment</small>	study_year	teaching_type <small>Type of teaching</small>	dac <small>Code of didactic activity</small>	department	curriculum	ssd <small>Scientific Disciplinary Sector (SSD)</small>
1	Analisi Matematica	9	2021/2022	1	Base	MFN0570	Informatica	Percorso generico	ANALISI MATEMATICA (MAT/05)
2	Architettura degli Elaboratori	9	2021/2022	1	Base	NULL	Informatica	Percorso generico	INFORMATICA (INF/01)
3	Agenti Intelligenti	6	2021/2022	1	Caratterizzante	MFN1348	Informatica	Intelligenza Artificiale e Sistemi	INFORMATICA (INF/01)

- **degree:** contiene le informazioni riguardanti i corsi di laurea.

degree_id **degree_name** **degree_cfu** **degree_description** **degree_type_id** **degree_location** **degree_athenaeum**

1	Informatica	180		1	Torino	Unito
2	Informatica	120		2	Torino	Unito
3	Matematica	180		1	Torino	Unito
4	Matematica	120		2	Torino	Unito
5	Fisica	180		1	Torino	Unito
6	Fisica	120		2	Torino	Unito

Chiave esterna:

- o **degree_type_id:** indica il tipo di corso di laurea (es. Triennale o Magistrale);

- **degree_type:** indica il tipo di corso di laurea.

degree_type_id **degree_type_name** **degree_type_note**

1	LT	Laurea Triennale
2	LM	Laurea Magistrale

- **degree_path:** contiene le informazioni riguardanti i curriculum.

Es. il percorso/curriculum "Realtà virtuale e Multimedialità", tramite chiave esterna degree_id, è associato al corso di laurea 2 (ovvero Corso di Laurea in Informatica Magistrale).

degree_path_id	degree_path_name	degree_path_description	degree_path_note	degree_id
1	Intelligenza Artificiale e Sistemi Informatici Pie...	Intelligenza Artificiale e Sistemi Informatici Pie...	per studenti iscritti dal 2017/2018	2
2	Sistemi per il Trattamento dell'Informazione	Sistemi per il Trattamento dell'Informazione	per studenti iscritti fino al 2016/2017	2
3	Reti e Sistemi informatici	Reti e Sistemi informatici	NULL	2
4	Realtà virtuale e Multimedialità	Realtà virtuale e Multimedialità	per gli studenti iscritti fino al 2019/2020	2
5	Immagini, Visione e Realtà Virtuale	Immagini, Visione e Realtà Virtuale	per studenti iscritti dal 2020/2021	2
6	Informazione e conoscenza	Informazione e conoscenza	NULL	1

- **course_path_degree:** tabella ternaria che mette insieme le informazioni riguardanti i corsi, le lauree e i percorsi/curriculum.

course_id **degree_path_id** **degree_id**

1	6	1
1	7	1
1	8	1

Es. Analisi Matematica (course_id = 1) è un corso relativo alla Laurea in Informatica (degree_id = 1) del curriculum Informazione e conoscenza (degree_path_id = 6).

- **tutor_course**: contiene le informazioni riguardanti i corsi insegnati dai tutor.
Es. il tutor con `user_id` = 11 (ovvero Paolo Rossi) effettua tutoraggio del corso con `course_id` = 3 (ovvero Agenti Intelligenti).

tutor_course_id	note	user_id	course_id
1	NULL	11	3
2	NULL	18	1
3	NULL	11	60
4	NULL	11	56

- **tutor_time_slot**: contiene le informazioni riguardanti le disponibilità dei tutor (giorni e orari).
L'attributo `user_id` è la chiave esterna che si riferisce all'utente (in questo caso il tutor).

tutor_time_slot_id	day	hour_from	hour_to	user_id
1	Martedì	18:00	21:00	11
2	Mercoledì	14:00	18:00	18
3	Giovedì	9:00	11:00	18
4	Venerdì	16:00	20:00	19

- **review**: contiene le informazioni riguardanti le recensioni che gli utenti (identificati dall'attributo `user_id`) effettuano sui tutor (identificati dall'attributo `user_tutor_id`).

review_id	user_tutor_id	user_id	review_star	review_comment
			star from 1 to 5	
1	11	1	5	Tutor eccellente!
2	11	2	2	Ottime spiegazioni ma l'esito dell'esame non è sta...

- **private_lesson**: contiene le informazioni sulle lezioni private degli studenti con riferimento al tutor e al corso prenotato.

private_lesson_id	user_id	tutor_course_id	tutor_time_slot_id	note
10	1	51	17	NULL
11	2	54	22	NULL
12	1	54	23	NULL
13	1	53	21	NULL
14	3	57	24	NULL

- **notifications_tutor**: contiene le notifiche ricevute dai tutor:
 - o **private_lesson_id** si riferisce alla lezione privata prenotata;
 - o **date** si riferisce alla data in cui è generata la notifica;
 - o **check** specifica se il tutor ha visto o meno la notifica (0 non vista, 1 vista).

notifications_tutor_id	private_lesson_id	date	check
1	23	2021-06-23 14:46:51	1
2	22	2021-06-21 00:00:00	1
3	25	2021-06-23 15:41:19	1
4	26	2021-06-23 16:05:54	1
5	27	2021-06-23 16:15:16	1
6	28	2021-06-23 16:15:16	1

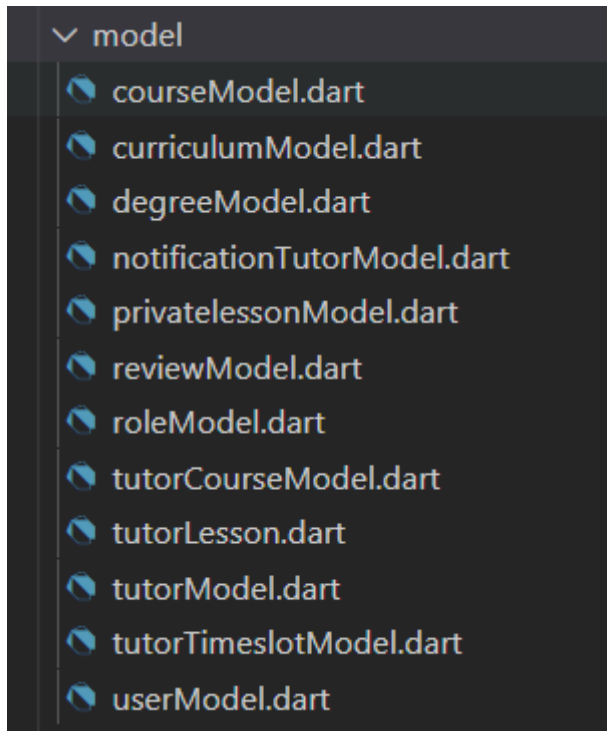
Web Service implementati

ADD	
add_private_lesson.php	Aggiunge una lezione privata insegnato da un tutor.
add_tutor_course.php	Aggiunge un corso alla lista dei corsi insegnati da un tutor.
add_tutor_time_slot.php	Aggiunge una disponibilità (time slot) ad un tutor.
add_user_review.php	Aggiunge una recensione di un utente (studente) ad un tutor.
user_signup.php	Registrazione di un nuovo utente (studente o tutor) all'applicazione.
DELETE	
user_delete.php	Elimina i dati dalle tabelle user e user_attribute dell'utente richiesto.
delete_tutor_course.php	Elimina un corso (course_id) dalla tabella tutor_course (corsi insegnati da un tutor).
delete_tutor_time_slot.php	Elimina una disponibilità (time_slot) dalla tabella time_slot.
UPDATE	
user_edit.php	Aggiorna i dati dalla tabella user_attribute dell'utente richiesto;
user_change_password.php	Prende in input e-mail e password, effettua l'encryption della password (md5) ed effettua l'update dei dati relativi all'utente identificato dall'email.
notifications_tutor_check.php	Setta l'attributo "check" a 1: indica che la notifica è stata vista dall'utente.
SELECT	
user_login.php	Verifica se l'e-mail e la password esistono nella tabella user: <ul style="list-style-type: none"> - se esistono, l'utente è abilitato e il login ha successo: restituisce "Login Matched"; - l'utente non è registrato nell'applicazione: restituisce "Invalid Username or Password Please Try Again".
notifications_tutor.php	Restituisce le notifiche relative alle prenotazioni degli studenti alle lezioni private offerte dal tutor. https://www.e-tutoring-app.it/ws/notifications_tutor.php?email=paolo.rossi@edu.unito.it
get_user_role.php	Restituisce il ruolo dell'utente (e-mail) passato come parametro al WS. https://www.e-tutoring-app.it/ws/get_user_role.php?email=paolo.rossi@edu.unito.it
course_list.php	Restituisce la lista di tutti i corsi. https://www.e-tutoring-app.it/ws/course_list.php Restituisce il singolo corso richiesto. https://www.e-tutoring-app.it/ws/course_list.php?course_id=1
course_search_private_lesson.php	https://www.e-tutoring-app.it/ws/course_search_private_lesson.php?email=luca.marignati@edu.unito.it Restituisce un array di lezioni private di uno specifico utente (identificato dall'email): es. Luca (user_id = 1) ha ricevuto una lezione di tutoraggio di Agenti Intelligenti da Paolo (user_tutor_id = 11)
course_search.php	https://www.e-tutoring-app.it/ws/course_search.php?email=luca.marignati@edu.unito.it Restituisce un array di corsi a cui l'utente è iscritto (corsi relativi al percorso di Laurea in cui l'utente è iscritto).
course_user_list.php	Restituisce un array di corsi relativi al percorso di Laurea in cui l'utente è iscritto.

	https://www.e-tutoring-app.it/ws/course_user_list.php?email=luca.marignati@edu.unito.it
curriculum_path_by_degree.php	https://www.e-tutoring-app.it/ws/curriculum_path_by_degree.php?degree_name=informatica&degree_type_note=Laurea%20Triennale Input: <ul style="list-style-type: none"> - nome del corso di laurea (es. informatica); - tipo del corso di laurea (es. triennale o magistrale); Output: array dei percorsi/curriculum disponibili per il corso di laurea richiesto.
degree_list.php	Restituisce un array che contiene la lista dei Corsi di Laurea disponibili. https://www.e-tutoring-app.it/ws/degree_list.php
degree_path_list.php	Array dei percorsi/curriculum dei corsi di laurea. https://www.e-tutoring-app.it/ws/degree_path_list.php Input: id del corso di laurea; Output: array dei percorsi/curriculum disponibili per il corso di laurea richiesto. https://www.e-tutoring-app.it/ws/degree_path_list.php?degree_id=1
degree_type_list.php	Restituisce un array contenente le tipologie di lauree disponibili (triennale o magistrale). https://www.e-tutoring-app.it/ws/degree_type_list.php
private_lesson_list.php	Restituisce un array contenente le lezioni private (tutoriali) ricevute da un utente e sostenute da un tutor. https://www.e-tutoring-app.it/ws/private_lesson_list.php
reviews_list.php	Restituisce un array contenente le recensioni ricevute dal tutor passato come parametro (user_tutor_id). https://www.e-tutoring-app.it/ws/reviews_list.php?user_tutor_id=11
role_list.php	Restituisce un array contenente la lista dei ruoli. https://www.e-tutoring-app.it/ws/role_list.php
tutor_list.php	Restituisce un array contenente la lista dei tutor (ovvero gli utenti aventi ruolo "tutor"). https://www.e-tutoring-app.it/ws/tutor_list.php Restituisce un JSON contenente i dati del singolo tutor passato come parametro (id). https://www.e-tutoring-app.it/ws/tutor_list.php?id=11 Restituisce un JSON contenente i dati del singolo tutor passato come parametro (e-mail). https://www.e-tutoring-app.it/ws/tutor_list.php?email=paolo.rossi@edu.unito.it
users_list.php	Restituisce un array contenente la lista degli utenti. https://www.e-tutoring-app.it/ws/users_list.php Restituisce un JSON contenente i dati di un singolo utente (id). Restituisce un JSON contenente i dati di un singolo utente (e-mail) https://www.e-tutoring-app.it/ws/users_list.php?email=luca.marignati@edu.unito.it

Model

Sono rappresentate le classi con i relativi campi corrispondenti a quelli restituiti dal WS (attributi delle tabelle del DB).



Es. couseModel.dart

```
28   CourseModel(  
29     this.course_id,  
30     this.course_name,  
31     this.course_cfu,  
32     this.enrollment_year,  
33     this.study_year,  
34     this.teaching_type,  
35     this.dac,  
36     this.department,  
37     this.curriculum,  
38     this.ssd,  
39     this.delivery_mode,  
40     this.language,  
41     this.didactic_period,  
42     this.component_type,  
43     this.private_lesson_id);  
44  
45   CourseModel.fromJson(dynamic json) {  
46     course_id = json['course_id'] ?? '-';  
47     course_name = json['course_name'] ?? '-';  
48     course_cfu = json['course_cfu'] ?? '-';  
49     enrollment_year = json['enrollment_year'] ?? '-';  
50     study_year = json['study_year'] ?? '-';  
51     teaching_type = json['teaching_type'] ?? '-';
```

Paradigma MVC

Abbiamo sviluppato l'applicazione adottando il paradigma MVC:

- **Model:** modello dei dati;
- **Controller:**
 - o si occupa di effettuare la chiamata http andando a chiamare i WS;
 - o mette a disposizione i dati al Widget (View);
- **View:** si occupa di rappresentare i dati forniti dal controller.

model	controller	screens
courseModel.dart	course_controllerWS.dart	calendar-student.dart
curriculumModel.dart	curriculum_controllerWS.dart	calendar-tutor.dart
degreeModel.dart	degree_controller.dart	change-password.dart
notificationTutorModel.dart	login_controllerWS.dart	course.dart
privateLessonModel.dart	notification_controllerWS.dart	courseDetail.dart
reviewModel.dart	private_lesson_controllerWS.dart	login.dart
roleModel.dart	review_controller.dart	my-tutor-course.dart
tutorCourseModel.dart	role_controllerWS.dart	my-tutor-lesson-today.dart
tutorLesson.dart	tutor_controllerWS.dart	my-tutor-lesson.dart
tutorModel.dart	tutor_lessonWS.dart	my-tutor-reviews.dart
tutorTimeslotModel.dart	user_controllerWS.dart	my-tutor-timeslot-add.dart
userModel.dart		my-tutor-timeslot.dart
		privacy-policy.dart
		profile-edit.dart
		profile.dart

Esempio 1:

- view: login.dart
 - o TextFormField email
 - o TextFormField password
 - o ElevatedButton login
 - o ElevatedButton signup
- Quando l'utente clicca sul bottone di login si scatena l'evento `onClicked`: viene chiamata la funzione `userLogin()` che gestisce la logica della funzionalità:
 - o entra in gioco il `login_controllerWS.dart` che fornisce la funzione `login()`: si occupa di chiamare il WS `user_login.php` e ritorna true o false in caso di successo o fallimento.

```
Future<bool> login(http.Client client, String email, String password) async {  
  var data = {'email': email, 'password': password};  
  // Starting Web API Call.  
  // https method: POST  
  var response = await client  
    .post(Uri.http(authority, unencodedPath + 'user_login.php'),  
      headers: <String, String>{'authorization': basicAuth},  
      body: json.encode(data))  
    .timeout(const Duration(seconds: 8));  
  
  if (response.statusCode == 200) {  
    var message = jsonDecode(response.body);  
  
    // If the Response Message is Matched.  
    if (message == 'Login Matched') {  
      return true;  
    } else {  
      return false;  
    }  
  } else {  
    return false;  
  }  
}
```

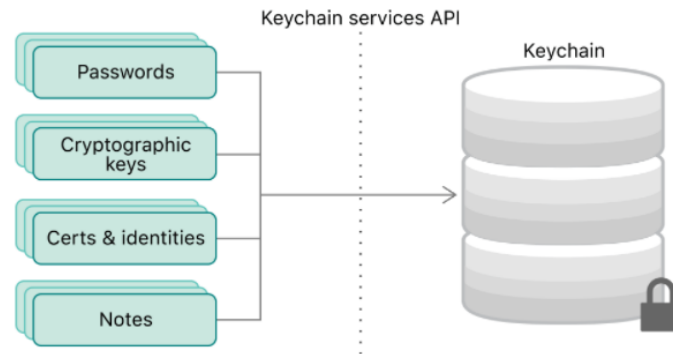
```
if (await login(http.Client(), email, password)) {  
  await UserSecureStorage.setEmail(emailController.text);  
  await UserSecureStorage.setPassword(passwordController.text);  
}
```

Principali librerie utilizzate e le possibili alternative

- **flutter_secure_storage: ^4.2.0**
https://pub.dev/packages/flutter_secure_storage
È un plug-in Flutter per archiviare i dati in maniera sicura.
- **flutter_localizations**
<https://flutter.dev/docs/development/accessibility-and-localization/internationalization>
- **intl: ^0.17.0**
<https://pub.dev/packages/intl>
- **http: ^0.13.3**
<https://pub.dev/packages/http>
Questo pacchetto contiene un insieme di funzioni e classi di alto livello che semplificano l'utilizzo delle risorse HTTP. È multiplatforma e supporta dispositivi mobili, desktop e browser.
- **flutter_local_notifications: ^5.0.0+4**
Un plug-in multiplatforma per la visualizzazione di notifiche locali.
https://pub.dev/packages/flutter_local_notifications
- **firebase_core: ^1.1.0**
Un plug-in Flutter per utilizzare l'API Firebase Core, che consente la connessione a più app Firebase.
https://pub.dev/packages/firebase_core
- **firebase_messaging: ^10.0.2**
Un plug-in Flutter per utilizzare l'API Firebase Cloud Messaging.
https://pub.dev/packages/firebase_messaging
<https://firebase.google.com/docs/cloud-messaging>

flutter_secure_storage

Concentriamoci descrivendo in dettaglio la libreria “flutter_secure_storage” discutendone i pro e i contro e analizzando le possibili alternative.



La libreria permette di **astrarre dalle primitive native** dei vari dispositivi fornendo dei metodi che permettono di scrivere (write) e leggere (read) le chiavi:

- **metodo write(key, value);**
- **metodo read(key);**
- **metodo delete(key).**

Nativamente:

- per iOS le chiavi vengono salvate in un archivio chiamato **Keychain** (https://developer.apple.com/documentation/security/keychain_services#/apple_ref/doc/uid/TP30000897-CH203-TP1);
- per Android, viene utilizzata la crittografia AES: la chiave segreta AES viene crittografata con RSA e viene archiviata nel **KeyStore** (<https://developer.android.com/training/articles/keystore>);
- per Linux viene utilizzata **libsecret** (<https://wiki.gnome.org/Projects/Libsecret>);

Nota: KeyStore è stato introdotto in Android 4.3 (livello API 18).

Il plugin non funzionerebbe per le versioni precedenti.

Scelta implementativa: **abbiamo utilizzato questa libreria per l'archiviazione sicura delle informazioni dell'utente (username, password e ruolo) nel dispositivo.**

Caratteristiche per Android (per IOS sono equivalenti):

- il Keystore fornisce un contenitore sicuro, che può essere utilizzato dalle applicazioni per memorizzare le chiavi private, in un modo che sia difficile per gli utenti malintenzionati (non autorizzati) recuperare le informazioni private;
- un'applicazione è in grado di memorizzare più chiavi nel Keystore, ma può solo visualizzare, e ricercare, le **sue chiavi**;
- il vantaggio di memorizzare una chiave nel Keystore è che consente di utilizzare le chiavi senza esporre il contenuto segreto di quella chiave: **i dati chiave non entrano nello spazio dell'app**. Le chiavi sono protette da permessi in modo che solo la tua app possa accedervi;
- vantaggi relativi alla sicurezza dell'accesso al Keystore e alle chiavi salvate.

Implementazione: è stata implementata la classe **UserSecureStorage** che si occupa di comunicare con il KeyStore (per Android) o il Keychain (per iOS). Per completezza, di seguito, abbiamo riportato il codice della classe.

```
1  import 'package:flutter_secure_storage/flutter_secure_storage.dart';
2
3  class UserSecureStorage {
4    static final _storage = FlutterSecureStorage();
5
6    static const _keyEmail = 'email';
7    static const _keyPassword = 'password';
8    static const _keyRole = 'role';
9
10   static Future setEmail(String email) async =>
11   |   await _storage.write(key: _keyEmail, value: email);
12
13   static Future<String> getEmail() async => await _storage.read(key: _keyEmail);
14
15   static Future setPassword(String password) async =>
16   |   await _storage.write(key: _keyPassword, value: password);
17
18   static Future<String> getPassword() async =>
19   |   await _storage.read(key: _keyPassword);
20
21   static Future setRole(String role) async =>
22   |   await _storage.write(key: _keyRole, value: role);
23
24   static Future<String> getRole() async => await _storage.read(key: _keyRole);
25
26   static void delete(String key) async => await _storage.delete(key: key);
27 }
28
```

Alternative: Secure storage / Shared Preferences / SQLite / Local File Storage

Le alternative a questo approccio potevano essere molteplici a partire dall'utilizzo delle Shared Preferences o di altre tipologie di Data Storage (es. SQL Database).

Brevemente analizziamo i pro e i contro dei vari approcci:

Shared Preferences	SQLite	Local File Storage
<p>Si tratta di un archivio chiave/valore in cui è possibile salvare un dato con una determinata chiave. Per leggere i dati dal negozio è necessario conoscere la chiave dei dati.</p> <p>Questo rende la lettura dei dati molto semplice.</p> <p>Un utilizzo tipico è la gestione delle preferenze utente (es. modalità scura dell'applicazione, ultimo tab usato, ecc.): questi sono piccoli dettagli che non richiedono oggetti di grandi dimensioni o database altamente strutturati e sono informazioni che non richiedono un'archiviazione sicura.</p> <p>Vantaggio principale: velocità nel reperimento del dato.</p>	<p>Adatto per il salvataggio di grandi quantità di stessi dati strutturati. Poiché i dati sono strutturati e gestiti dal database, possono essere interrogati per ottenere un sottoinsieme dei dati che corrisponde a determinati criteri utilizzando un linguaggio di query come SQL.</p> <p>Ciò rende possibile la ricerca nei dati.</p>	<p>È una modalità semplice nella quale andiamo a creare un file, tipicamente un file di testo, nel quale possiamo scrivere e dal quale possiamo leggere alcuni dati. Di default questo tipo di salvataggio è un salvataggio privato e ciò implica che il file può essere letto e scritto esclusivamente dall'applicazione che lo ha creato e non è dunque possibile accedervi da un'altra applicazione.</p>
<p>Problemi di spazio: i dispositivi hanno dei limiti di memoria e il sistema potrebbe eliminare le shared preferences se occupano molto spazio: questo significa che nessuna delle preferenze condivise dovrebbe essere una funzionalità vitale dell'app.</p> <p>Dati non sicuri: è facile recuperare le informazioni da parte di utenti malintenzionati. Proprio per questo non devono essere memorizzate informazioni private.</p> <p>È difficile archiviare e leggere dati strutturati di grandi dimensioni in quanto è necessario definire la chiave per ogni singolo dato. Inoltre, non è possibile eseguire ricerche all'interno dei dati se non si ha un certo concetto per nominare le chiavi.</p>	<p>Ovviamente la gestione e la ricerca di grandi insiemi di dati influenzano le prestazioni, quindi la lettura dei dati da un database può essere più lenta rispetto alla lettura dei dati da Shared Preferences.</p>	<p>È difficile archiviare e leggere dati strutturati di grandi dimensioni in quanto è necessario leggere l'intero file. Inoltre, non è possibile eseguire ricerche.</p>

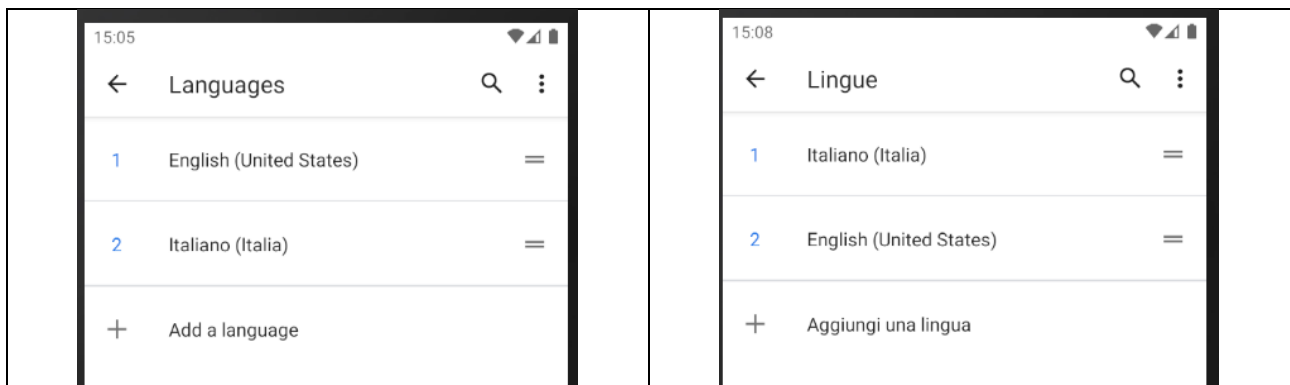
Flutter e supporto per device multipli

L'applicazione è stata sviluppando mediante l'utilizzo di Flutter e del linguaggio Dart fruttandone i numerosi vantaggi che forniscono:

- **cross-platform e indipendenza dal sistema operativo** (un codice, più piattaforme): le applicazioni sviluppate in Flutter implementano uno strato intermedio definito Bridge che permette di comunicare con le API native dei vari dispositivi (Android e iOS). È, dunque, possibile scrivere il codice una sola volta e non differenziare per le diverse tipologie di SO;
- **performance simili a quelle native**: nonostante la presenza dello strato di Bridge che comunica con le API native, le performance non vengono impattate negativamente e sono simili a quelle delle App native;
- **hot reloading – impatto positivo sulle tempistiche di sviluppo**: la funzione hot reload aiuta a sperimentare rapidamente e facilmente, creare interfacce utente, aggiungere funzionalità e correggere bug più velocemente. Stimola il processo di sviluppo fornendo a uno sviluppatore un record del codice sorgente direttamente all'interno dell'applicazione funzionante. Aiuta a riflettere sulle modifiche apportate al codice in meno di 2-3 secondi e senza ripristinare lo stato dell'applicazione. Pertanto, Hot Reload consente di monitorare rapidamente lo sviluppo dell'applicazione. A seconda della complessità e della natura del progetto, è possibile risparmiare dal 20 al 50% in termini di tempo, che può essere speso per lo sviluppo di altre funzioni utili.
- **semplicità nell'implementazione della logica**: Flutter fornisce funzionalità del sistema operativo avanzate come coordinate GPS, raccolta dei dati dei sensori, gestione delle autorizzazioni, Bluetooth, credenziali e altre funzionalità in **plug-in pronti per l'uso supportati da Google**. Se la app che progettate di sviluppare fa affidamento su una funzionalità a livello di sistema operativo non disponibile come plug-in, Flutter può stabilire la comunicazione tra il suo linguaggio di programmazione Dart e il codice nativo utilizzando i canali della piattaforma.
- **riduzione dei costi di sviluppo**: con Flutter si riducono notevolmente i costi di sviluppo dell'app, visto che si utilizza lo stesso set di librerie per l'interfaccia utente e il medesimo framework. Flutter è in grado di gestire ogni singolo pixel del display, in questo modo si raggiungeranno altissimi livelli di personalizzazione e si potranno sviluppare interfacce utente con grafiche innovative e dettagli visivi di altissima qualità;
- **framework open source**: si è creata una vera e propria community di sviluppatori, che pubblicano esempi di codice e che aiutano e supportano gli utenti nella creazione di nuove app cross-platform.

Supporto per le lingue diverse

È possibile impostare la lingua del dispositivo andando su “Settings” → “Languages” (es. su Android) e portando la lingua scelta in cima alla lista (vedi figura sotto).



L'applicazione, in base, a questa configurazione adatterà i propri contenuti in base alla lingua scelta sfruttando le librerie *flutter_localizations* e *intl* che forniscono le funzionalità di internazionalizzazione, localizzazione e la traduzione dei messaggi.

Lingue supportate dell'applicazione: **inglese e italiano**.

Sarebbe triviale aggiungere il supporto di un'altra lingua:

- la cartella “l10n” contiene i file di traduzioni in formato ARB in cui le risorse sono codificate come oggetti JSON;
- in caso si voglia aggiungere il supporto di una terza lingua basterebbe aggiungere il relativo file che contiene le traduzioni per la nuova lingua;

In questi file vengono specificate **coppie CHIAVE-VALORE**. Es:

- inglese: chiave “welcome”, valore “welcome to\nE-Tutoring”;
- italiano: chiave “welcome”, valore “Benvenuto in\nE-Tutoring”;

app_en.arb	app_it.arb
<pre>"welcome": "Welcome to\nE-Tutoring", "@welcome": { "description": "Welcome to E-Tutoring" }, "login": "Login", "@login": { "description": "Login" }, "signup": "Sign up", "@signup": { "description": "Sign up" }, "yourpassword": "Your Password", "@yourpassword": { "description": "Your Password" }, "youremail": "Your Email"</pre>	<pre>"welcome": "Benvenuto in\nE-Tutoring", "login": "Accedi", "signup": "Registrati", "yourpassword": "Inserisci la password", "youremail": "Inserisci l'email", "error_email_empty": "Inserisci la tua email", "error_email_not_valid": "Inserisci un email valida", "error_password_empty": "Inserisci la tua password", "error_password_not_valid": "Inserisci una password", "confirmPassword": "Conferma Password", "passwords_not_match": "Le password inserite non comb", "select_degree_course": "Scegli il corso di laurea",</pre>

FLUTTER GENERATE TRUE: evidenziamo il comportamento della libreria: in fase di compilazione dell'applicazione, vengono generate le seguenti classi:

- **.dart_tool/flutter_gen/genl10_n/app_localizations.dart**: classe astratta che contiene le chiavi di traduzione;
- **.dart_tool/flutter_gen/genl10_n/app_localizationsIt.dart**: classe che espone i metodi get delle chiavi fornendo il valore della relativa traduzione in lingua italiana;
- **.dart_tool/flutter_gen/genl10_n/app_localizationsEn.dart**: classe che espone i metodi get delle chiavi fornendo il valore della relativa traduzione in lingua inglese;

app_localizationsIt.dart	app_localizationsEn.dart
<pre> @override String get welcome => 'Benvenuto in\nE-Tutoring'; @override String get login => 'Accedi'; @override String get signup => 'Registrati'; @override String get yourpassword => 'Inserisci la password'; @override String get youremail => 'Inserisci l\'email'; @override String get error_email_empty => 'Inserisci la tua email'; @override String get error_email_not_valid => 'Inserisci un email valida'; </pre>	<pre> @override String get welcome => 'Welcome to\nE-Tutoring'; @override String get login => 'Login'; @override String get signup => 'Sign up'; @override String get yourpassword => 'Your Password'; @override String get youremail => 'Your Email'; @override String get error_email_empty => 'Please enter your email'; @override String get error_email_not_valid => 'Please enter a valid email'; </pre>

CONFIGURAZIONE

```

@override
Widget build(BuildContext context) {
  final provider = Provider.of<LocaleProvider>(context);
  return MaterialApp(
    debugShowCheckedModeBanner: false,
    locale: provider.locale,
    supportedLocales: L10n.all,
    localizationsDelegates: [
      AppLocalizations.delegate,
      GlobalMaterialLocalizations.delegate,
      GlobalCupertinoLocalizations.delegate,
      GlobalWidgetsLocalizations.delegate,
    ],
    home: _body,
    onGenerateRoute: RouteGenerator.generateRoute,
  );
}

```

UTILIZZO (tramite AppLocalizations): si definisce la chiave della traduzione (es. login):

```

appBar: AppBar(
  title: Text("E-Tutoring " + AppLocalizations.of(context).login),
  backgroundColor: Color.fromRGBO(213, 21, 36, 1),
  actions: [LanguagePickerWidget()],
), // AppBar
backgroundColor: Colors.white,
body: Scaffold(

```

Test effettuati

Abbiamo sviluppato i test suddividendo in cartelle rispettando la struttura del codice in modo da avere un'esatta corrispondenza tra implementazione e test:

- nella cartella di `test/controller` abbiamo testato le funzioni relative a `lib/controller` (in particolare le chiamate http);
- nella cartella `test/screens` abbiamo testato le funzioni relative a `lib/screen`;
- nella cartella `test/utils` abbiamo testato le funzioni relative a `lib/utills`;
- nella cartella `test/widgets` abbiamo testato le funzioni relative a `lib/widgets`.

```
C:\Users\luca\Desktop\ETutoringFlutter>flutter test test\
00:12 +64: All tests passed!
```

test/controller - Mockito

Utilizzando la libreria **Mockito** abbiamo testato i metodi relativi alle chiamate http.

<https://flutter.dev/docs/cookbook/testing/unit/mocking>

Per ogni metodo del controller abbiamo implementato un test contenente 2 casi:

- risposta HTTP con status 200: operazione va a buon fine;
- risposta HTTP con status 404 (NOT FOUND): operazione fallita;

Mostriamo di seguito un esempio di test spiegando i passi implementativi del test preso in esame: per esempio, vogliamo testare il metodo `getCurriculumListFromWS` che va chiama il WS specifico per fornire la lista dei curriculum di un corso di laurea (es. informatica, laurea triennale).

```
6
7 Future<List<CurriculumModel>> getCurriculumListFromWS(
8   http.Client client, String degreeName, String degreeTypeName) async {
9   List<CurriculumModel> curriculumList = [];
10
11   //https://www.e-tutoring-app.it/ws/curriculum_path_by_degree.php?degree_name=
12   var queryParameters = {
13     'degree_name': degreeName,
14     'degree_type_note': degreeTypeName
15   };
16   try {
17     var response = await client.get(
18       Uri.https(authority, unencodedPath + "curriculum_path_by_degree.php",
19         queryParameters),
20       headers: <String, String>{'authorization': basicAuth});
21     if (response.statusCode == 200) {
22       var curriculumJsonData = json.decode(response.body);
23       for (var curriculumItem in curriculumJsonData) {
24         var curriculum = CurriculumModel.fromJson(curriculumItem);
25         curriculumList.add(curriculum);
26       }
27     }
28     return curriculumList;
29   } on Exception catch (e) {
30     print('error caught: ' + e.toString());
31     return [];
32   }
33 }
34
```

https://www.e-tutoring-app.it/ws/curriculum_path_by_degree.php?degree_name=informatica°ree_type_note=Laurea%20Triennale

```
1  [
2    {
3      "degree_path_name": "Informazione e conoscenza"
4    },
5    {
6      "degree_path_name": "Linguaggio e sistemi"
7    },
8    {
9      "degree_path_name": "Reti e sistemi informatici"
10   }
11 ]
12
13
14
```

TEST: **inizializzazione del MockClient()** che permette di simulare il comportamento di un client http (browser);

1. **definizione dei parametri richiesti dal WS:** in questo caso *degree_name* e *degree_type_note*;
 2. simulazione della chiamata http utilizzando il mock client: abbiamo simulato la chiamata http;
 3. trattandosi di una chiamata fittizia abbiamo dovuto definire la risposta fornita dal WS in formato JSON: `[{"degree_path_name": "Immagini, Visione e Realtà Virtuale"}]` e definito lo status (es. 200 o 404);
 4. abbiamo chiamato la funzione implementata (*getCurriculumFromWS*);
 5. infine, abbiamo eseguito i vari test (es. test sul tipo ritorno) in modo da verificare che ci aspettiamo come risposta sia quello ottenuto dalla funzione e dal WS.
- Per esempio, ci aspettiamo che la funzione *getCurriculumFromWS* restituisca una lista di oggetti di tipo *CurriculumModel*.

Di seguito mostriamo il codice relativo al test.

```
15
16  ✓ getCurriculumListFromWSTest() {
17    Run | Debug
18    group('getCurriculumListFromWS', () {
19      Run | Debug
20      test(
21        'return a List of CurriculumModel if the http call completes successfully',
22        () async {
23          final client = MockClient();
24          var queryParameters = {
25            'degree_name': 'informatica',
26            'degree_type_note': 'Laurea Magistrale'
27          };
28          when(client.get(
29            Uri.https(
30              authority,
31              unencodedPath + "curriculum_path_by_degree.php",
32              queryParameters),
33              headers: <String, String>{'authorization': basicAuth}))
34            .thenAnswer((_) async => http.Response(
35              '[{"degree_path_name": "Immagini, Visione e Realtà Virtuale"}]',
36              200));
37
38          List<CurriculumModel> curriculumList = await getCurriculumListFromWS(
39            client, 'informatica', 'Laurea Magistrale');
40          expect(curriculumList, isA<List<CurriculumModel>>());
41        });
42      });
43    });
44  }
```

Gli altri test di questo tipo sono del tutto similari.
Per i dettagli rimandiamo al codice.

test/screens

Questi test riguardano gli screens ChangePassword, Login e Signup.

Come si nota dalle figure sotto, oltre ai test relative alle chiamate http (mockito), abbiamo implementato degli UNIT TEST:

- per esempio, abbiamo testato delle funzioni a livello unitario che riguardano la validazione dell'e-mail inserita dall'utente;
- abbiamo testato le funzioni che lavorano sui tipi di date. Per esempio la funzione formatDate prende in input una stringa, effettua il parsing in un oggetto di tipo DateTime ed rielabora la data nel formato dd-mm-aaa;

```
Run | Debug
test('empty email return error', () async {
  // unit test
  var result = EmailValidator.validate('');
  expect(false, result);
});

Run | Debug
test('not email return error', () async {
  // unit test
  var result = EmailValidator.validate('email not valid');
  expect(false, result);
});

Run | Debug
test('email returns true', () async {
  // unit test
  var result = EmailValidator.validate('paolo.rossi@edu.unito.it');
  expect(true, result);
});

Run | Debug
test('formatDate return a date in format string dd-MM-yyyy', () async {
  DateTime date = DateTime.parse('2021-05-28');
  expect(formatDate(date), "28-05-2021");
});

Run | Debug
test('Date Format EEEE return a string day of the week', () async {
  expect(DateFormat('EEEE').format(DateTime.parse("2021-05-28")), "Friday");
});

Run | Debug
test('Event class with title passed as argument', () async {
  Event event = new Event("test event");
  expect(event.title, "test event");
});
```

Mockito

Es. testing del metodo di login (WS: user_login.php). Nel test riportato ci aspettiamo che il metodo di login restituisca il booleano "false" in quanto username (simone.bortolotti@edu.unito.it) e password ("errore") non sono corrette, ovvero non esiste un match nel DB (nella tabella "user");

```
Run | Debug
test('login: verify email and password: false (not match)', () async {
  final client = MockClient();

  String email = "simone.bortolotti@edu.unito.it";
  String password = "errore";
  var data = {'email': email, 'password': password};

  when(client.post(Uri.https(authority, unencodedPath + 'user_login.php'),
    headers: <String, String>{'authorization': basicAuth},
    body: json.encode(data)))
    .thenAnswer((_) async => http.Response(
      jsonEncode('Invalid Username or Password Please Try Again'), 200));

  bool loginResult = await login(client, email, password);
  expect(loginResult, false);
});
```